**Git**

**Cours** :

<http://www.siteduzero.com/tutoriel-3-254198-gerez-vos-codes-source-avec-git.html>

<http://www.alexgirard.com/git-book/index.html>

**But** :

Un logiciel de gestion de version (catégorie : distribuée) est principalement utilisé pour gérer et suivre l’évolution des fichiers d’un code source, tout en gardant leurs anciennes versions. Ils sont conseillés pour gérer les projets informatiques.

* Stocke en plus qui a fait la modification, pourquoi et quand,
* Permet de travailler sans conflit à plusieurs sur un même projet (même fichier).

[Il existe d’autres logiciels de gestion de versions.](http://www.siteduzero.com/tutoriel-3-254198-gerez-vos-codes-source-avec-git.html#ss_part_2)

**Mode de fonctionnement – Distribué + serveur**



**Installation de GIT sous Windows**

Site : <http://code.google.com/p/msysgit/downloads/list>

Note : Cela installe msys (système d’émulation des commandes Unix sous Windows) et Git simultanément.

Protection : [créer une paire de clés avec Puttygen](http://www.siteduzero.com/tutoriel-3-254198-gerez-vos-codes-source-avec-git.html#ss_part_2)

**Configuration générale de Git**

|  |  |
| --- | --- |
| git config –-global user.name ”votre\_pseudo” | Configurer le pseudo utilisé pour les commit |
| git config –-global user.email [moi@email.com](mailto:moi@email.com) | Configurer l’email utilisé pour les commit |
| git config –-global color.diff auto | Active la couleur dans git |
| git config –-global color.status auto | Active la couleur dans git |
| git config –-global color.branch auto | Active la couleur dans git |

Editer le fichier de configuration .gitconfig (dans le répertoire personnel)

|  |  |
| --- | --- |
| vim ~/.gitconfig | [Edition du fichier configuration](http://doc.ubuntu-fr.org/vim) |
| [alias]  ci = commit  co = checkout  st = status  br = branch |  |

**Configuration propre à un projet**

|  |  |
| --- | --- |
| git config user.name ”votre\_pseudo” | Configurer le pseudo utilisé pour les commits du projet |
| git config user.email [moi@email.com](mailto:moi@email.com) | Configurer l’email utilisé pour les commits du projet |

**Les dépôts**

Dans les logiciels de gestion de version, un dépôt est la copie d’un projet sur l’ordinateur d’un développeur travaillant dessus. Dans chaque dépôt, on trouve les fichiers du projet ainsi que leur historique.

**Créer un nouveau dépôt ou cloner un dépôt existant**

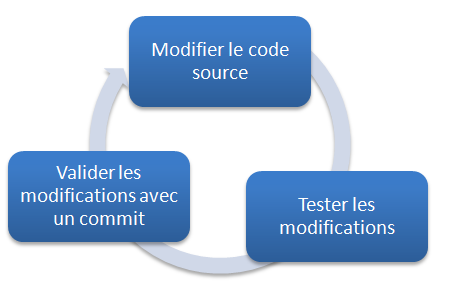
1 – Création d’un dépôt vide pour commencer un nouveau projet

|  |  |
| --- | --- |
| cd /dans/dossier/projet | Positionnement dans le dossier du projet (même vide) |
| git init | Initialise un nouveau dépôt vide Git dans le dossier. |

2 – Clonage d’un dépôt existant : récupération de tout l’historique les changements d’un projet pour travailler dessus.

|  |  |
| --- | --- |
| cd /dans/dossier/projet | Positionnement dans le dossier du projet (même vide) |
| git clone http://github.com/symfony/symfony.git | Clone le dépôt de Symfony dans le dossier courant de l’ordinateur. |

**Méthode de travail avec Git**



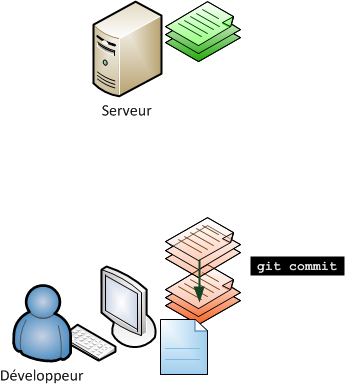
3

2

1

Un commit permet d’enregistrer localement et officiellement dans l’historique vos modifications effectuées sur le projet.

**Note :** Une demande redmine corrigée et testée = un commit



**Connaître les statuts d’un dépôt**

|  |  |
| --- | --- |
| cd /dans/dossier/projet | Positionnement dans le dossier du projet avec un dépôt |
| git status | Indique les fichiers modifiés et récemment commiter |

* Aucune modification effectuée sur les fichiers du projet

$ git status

# On branch master

nothing to commit (working directory clean)

* Modification et enregistrement d’un fichier du projet

$ git status

# On branch master

# Changed but not updated:

# (use "git add <file>…" to update what will be committed)

# (use "git checkout -- <file>…" to discard changes in working directory)

#

# modified: src/Symfony/Components/Yaml/Yaml.php

#

no changes added to commit (use "git add" and/or "git commit -a")

**Connaître concrètement les changements effectués**

|  |  |
| --- | --- |
| cd /dans/dossier/projet | Positionnement dans le dossier du projet avec un dépôt |
| git diff | Affiche les différences |

$ git diff

diff --git a/src/Symfony/Components/Yaml/Yaml.php b/src/Symfony/Components/Yaml/

index fa0b806..77f9902 100644

--- a/src/Symfony/Components/Yaml/Yaml.php

+++ b/src/Symfony/Components/Yaml/Yaml.php

@@ -19,7 +19,7 @@ namespace Symfony\Components\Yaml;

\*/

class Yaml

{

- static protected $spec = ’1.2’;

+ static protected $spec = ’1.3’;

/\*\*

\* Sets the YAML specification version to use.

@@ -33,6 +33,8 @@ class Yaml

if (!in\_array($version, array(’1.1’, ’1.2’))) {

throw new \InvalidArgumentException(sprintf(’Version %s of the YAML

}

+

+ $mtsource = $version;

self::$spec = $version;

}

**+** Ligne ajoutée

**-** Ligne supprimée

|  |  |
| --- | --- |
| git diff nom/fichier/unique.php | Affiche les différences |

**Effectuer un commit des changements**

|  |  |
| --- | --- |
| git status | Affiche en rouge les fichiers modifiés.  Note : Ils ne seront pas pris en compte lors du prochain commit. |

Il faut préciser quels fichiers à assembler pour être ensuite commiter. 3 possibilités :

Cas 1

|  |  |
| --- | --- |
| git add nomfichier1 nomfichier2 | Ajoute les fichiers à la liste de ceux qui peuvent être commiter. |
| git status | Les fichiers ajoutés à la liste à commiter s’affiche en vert |
| git commit | Effectue le commit.  Note : l’éditeur par défaut s’ouvre et sur la première ligne, il faut saisir la description des modifications faites. Ce message est obligatoire. |

Cas 2 (la préférée)

|  |  |
| --- | --- |
| git commit -a | Commite tous les fichiers qui s’affichent en vert ou en rouge après un git status |

Cas 3

|  |  |
| --- | --- |
| git commit nomfichier1 nomfichier2 | Effectue le commit des fichiers indiqués |

- Ignorer des fichiers

Ces fichiers ne seront pas intégrés dans les traitements de Git (par exemple : les fichiers temporaires d’un projet).

Création d’un fichier nommé .gitignore à la racine du répertoire de travail.

# Les ligne commençant par '#' sont des commentaires.

# Ignorer tous les fichiers nommés foo.txt

foo.txt

# Ignorer tous les fichiers html

\*.html

# à l'exception de foo.html qui est maintenu à la main

!foo.html

# Ignorer les objets et les archives

\*.[oa]

**Annuler un commit effectué par erreur**

- Consulter l’historique des commits : les logs.

|  |  |
| --- | --- |
| git log | Permet de retrouver toutes les modifications depuis le début du projet.  « Page up » et « Page down » « les flèches » et « Q » pour quitter. |
| git log -p | Détail des lignes qui ont été ajoutées et retirées dans chaque commit. |
| git log –-stat | Résumé plus court des commits. |

- Corriger une erreur non-committée

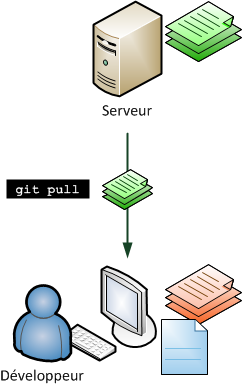
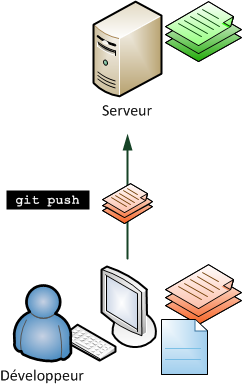
|  |  |
| --- | --- |
| git reset --hard HEAD | Permet de retrouver l’état dans lequel était le répertoire après le dernier commit. |
| git checkout -- nomfichier | Restaure le fichier à la version de l’index après le dernier commit. |
| git checkout HEAD nomfichier | Annule les modifications du fichier avant un commit. |

- Corriger une erreur commitée

On doit créer un nouveau commit qui annule les changements du dernier commit.

|  |  |
| --- | --- |
| git commit –amend | Permet de modifier le dernier message de commit. |

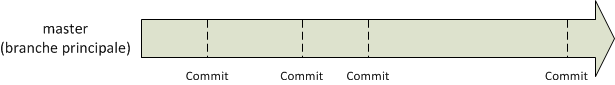
**Télécharger les nouveautés et partager votre travail**

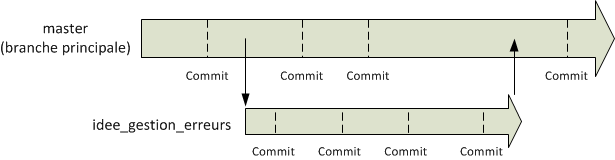
|  |  |
| --- | --- |
| git pull | Télécharge les nouveautés depuis le serveur et les fusionne à celles faites en local |
| git push | Envoie les modifications faites en local sur le serveur (serveur du clone)  Note : Avant l’envoi, vérifier une dernière fois les logs locaux avec un git log -p |

**Travail avec les branches**

- Master : la branche principale



- Sous-branche



*« idee\_gestion\_erreurs »* : exemple d’une sous-branche

Note : ne pas hésiter à créer des branches lors de modifications qui peuvent être soit longues, soit compliquées, soit qu’elles nécessitent plusieurs commit . Ainsi, on peut corriger des bugs mineurs sur le master et livrer régulièrement des versions corrigées aux clients même pendant la durée de la modification principale.

**Les branches locales**

|  |  |
| --- | --- |
| git branch | Affiche toutes les branches d’un projet.  Notes : L’astérisque « \* » devant la branche indique celle sur laquelle, on est en train de travailler. Par défaut, le master |
| git branch nom\_de\_la\_branche | Crée une branche locale nommée « nom\_de\_la\_branche » |
| git checkout nom\_de\_la\_branche | Permet de se positionner sur la branche locale nommée « nom\_de\_la\_branche »  Notes : C’est cette branche qui prend l’astérique « \* » |
| git checkout master | Permet de se positionner sur la branche principale. |
| git merge nom\_de\_la\_branche | Fusionne les modifications de la branche indiquée vers la branche master. |
| git branch –d nom\_de\_la\_branche | Supprime la branche indiquée après en avoir fusionné les modifications dans master. |
| git branch –D nom\_de\_la\_branche | Supprime la branche indiquée et perd tous les changements. |

**Annuler un merge**

|  |  |
| --- | --- |
| git reset --hard HEAD | Annule le dernier merge (non commiter) |
| git reset --hard ORIG\_HEAD | Annule le dernier merge (déjà commiter) |

**Les branches partagées**

|  |  |
| --- | --- |
| git branch –r | Liste toutes les branches que le serveur connaît. |
| git branch –-track nouvelle\_branche origin/nouvelle\_branche | Crée un nouvelle\_branche qui sera connecté à la nouvelle\_branche sur le serveur.  Note : remote tracking branches |

**Ajouter ou supprimer une branche sur le serveur**

|  |  |
| --- | --- |
| git push origin origin :refs/heads/nom\_nouvelle\_branche | Ajoute la branche sur le serveur pour y travailler à plusieurs. |
| git push origin :heads/nom\_branche | Supprime la branche indiquée sur le serveur. |
| git branch –r –d origin/nom\_branche | Les clients doivent supprimer les remote tracking branches |

**Les tags**

- Tag léger

|  |  |
| --- | --- |
| git tab NomTag IDCommit | Donne un alias à un commit précis pour le référencer sous ce nom. |
| git tag v1.3 2f7c8b3428aca535fdc970feeb4c09efa33d809e | 🡸 Exemple |
| git push - -tags | Précise qu’il faut envoyer les tags au moment d’un push |
| git tag –d NomTag | Supprime le tag indiqué. |

- Les objets tag

|  |  |
| --- | --- |
| git tab -a NomTag IDCommit | Crée un objet tag qui pointe sur un commit précis pour le référencer sous ce nom.  Note : Un éditeur se lance pour saisir le message du tag. |
| git tag v1.3 2f7c8b3428aca535fdc970feeb4c09efa33d809e | 🡸 Exemple |